BACKGROUND AND ARCHITECTURE FOR AN AUTONOMOUS GROUND STATION CONTROLLER.

Leslie Paal
Jet Propulsion Laboratory, Communication Systems and Research Section
Forest Fisher
Jet Propulsion Laboratory, Information Technologies and Software Systems Section
Mark James
Jet Propulsion Laboratory, Information Technologies and Software Systems Section
Dr. Nasser Golshan
Jet Propulsion Laboratory, Communication Systems and Research Section

The Deep Space Station Controller (DSSC) is state of the art ground station control architecture being developed at the Jet Propulsion Laboratory (JPL). During the past few years the technology development program at JPL demonstrated a series of increasingly competent automated ground station prototypes of which the DSSC is the latest. It has been designed for robust closed loop control of ground stations utilized for forward and return link communications with NASA's deep space exploration missions.

1. INTRODUCTION

The NASA Office of Space Communications conceived the idea of an automated, unattended terminal to reduce the life-cycle cost of tracking near earth missions. JPL was tasked in FY 1994 to conduct a demonstration of the concept. Successful demonstrations of the automated, unattended operations of the Low Earth Orbiter Demonstration (LEO-D) station were conducted. Next the station was modified to add uplink ability for full service (telemetry and command) support of LEO missions; renamed as LEO Terminal (LEO-T). In 1996, the Deep Space Terminal (DS-T) task began to demonstrate the proof of concept of a fully automated and autonomous "lights-out" ground station in the Deep Space Network (DSN) [1] environment. A series of successful "lights-out" demonstrations were conducted. The lessons learned from the DS-T influenced the team to change the automation approach. The Deep Space Station Controller (DSSC) has been designed with a powerful Fault Detection and Isolation (FDI) function feeding a Planner that reassesses the next action which moves the system to the desired goal.

Figure 1 70-Meter Deep Space Communication Antenna

To better understand the difficulties that the DSSC has to overcome, a short introduction to the DSN and the operating conditions is helpful (Section 2). The milestones achieved and significant lessons learned during the LEO-D [3], LEO-T and DS-T projects [4] are described in Section 3. To address these lessons we are developing a new modular and extendable architecture for the DSSC (Section 4). This new architecture would replace the existing system that has been enhanced numerous times in the past resulting in a very complex, cumbersome system [2]. While initially this architecture is targeted for station controller, it was designed so that the same architecture and much of the same code can be used as a complex controller or a subsystem controller.

The DSSC is built around two powerful state of the art technologies developed through JPL's Telecommunications Mission Operations Directorate (TMOD) technology program. These are: 1) a reasoning and controlling component, CLEaR (Closed Loop Execution and Recovery) for selecting, issuing and monitoring DSN component commands as well as re-planning recovery scenarios (Section 5). 2) A two-component fault detection and isolation technology, Beacon-based Exception Analysis for Multi-missions (BEAM) and Spacecraft Health Inference Engine (SHINE) (Section 6). BEAM is used as a highly advanced prognostic state estimator and SHINE is used for hard real-time diagnostics and interpretation of the BEAM state. The combination of the Planner and FDI technologies enables the system to perform both intelligent understanding and intelligent reasoning.

2. What the DSN is and how it Operates

In order to collect both science and engineering data from spacecraft, NASA operates a network of communication antennas called the Deep Space Network. The DSN was established in 1958 and since then it has evolved into the largest and most precise scientific telecommunications and radio navigation network in the world. The primary purpose of the DSN is to support interplanetary spacecraft missions and support radio and radar astronomy observations in the exploration of the solar system and the universe. For many near earth spacecraft, the DSN also performs an important backup function. The DSN currently consists of three deep-space communication complexes placed approximately 120 degrees apart around the world: at Goldstone, USA, near Madrid, Spain, and near Canberra, Australia. Each DSN complex operates a set of deep space stations consisting of: one 70-meter antenna (Figure 1), a collection of 34-meter antennas, one 26-meter antenna, and 11-meter antennas. The functions of the DSN are to receive telemetry data from spacecraft, transmit commands that control the spacecraft actions, generate the radio navigation data used to locate and guide the spacecraft to its destination. In addition the DSN also collects flight radio science, radio and radar astronomy, very long baseline interferometry, and geo-dynamics measurements.

The operation of the DSN is a very difficult task due to the extreme sensitivity of the equipment, the volume of data collected, the range of missions operated, and the frequency at which service must be provided. Deep Space missions present a unique environment; the extreme distance means very low signal levels. Communicating with an Outer Planet mission to Neptune or Pluto requires equipment *~10 billion* times more sensitive than for a commercial GEO satellite. Long Round trip Light Times (up to hours) require different communication protocols. The DSN provides service to nearly 50 different missions; 24X7 operations, 52 weeks per year. On average over 350 communications tracks performed each week by the DSN where each tracking session has varying configuration and requirements. Several different frequencies are supported (L, S, X, Ka, Ku bands); data rates vary from a few b/s to over 2 Mb/s and several forward error correction methods are used. The majority of tracks last about 10 hours however some can be as short as 15 minutes. Most activities are planned well in advance, but emergency support is sometimes needed. While most missions follow the Consultative Committee for Space Data Systems (CCSDS) standards, many of the missions from yesteryear are still exploring the solar system and beyond, like Voyager I & II.

The first step in performing a DSN track is called network preparation. Here, a project sends a request for the DSN to track a spacecraft involving specific tracking services (e.g. downlink, uplink, ranging). The DSN responds by attempting to schedule the necessary resources (antenna, other shared equipment and support information) needed for the track. Once all necessary components are in place, the next step is the actual track, which is conducted by operations personnel at a DSN station. During this process, operators execute the appropriate steps to perform the needed tasks: configure the equipment for the track, establish the communications links (ground and space), and then perform the actual track, all by issuing control commands to the various subsystems comprising the link. Throughout the track the operators continually monitor the status of the communications link and handle exceptions (e.g. the receiver looses signal lock) as they occur. To perform all of these actions, human operators manually issue tens to hundreds of directives via a computer terminal.

In an attempt to reduce cost and increase operations reliability, the DSN considered automation. At the same time, the DSN has to do more with less. Operations budgets continue to shrink while NASA continues to launch more missions requiring service; trend to more and smaller missions: several new ones per year. Small missions are often more challenging, lower power and smaller staff. The goals for the automated system are several fold: (1) Missions efficient utilization of their allotted tracking slots, more reliable and uninterrupted operations; (2) Operations ability to respond quickly to real-time mission-critical faults and (3) Maximization of DSN equipment utilization by autonomous operations which reduces lifecycle costs. This paper discusses the DSSC being developed as a prototype station controller to automate a significant portion of these operational tasks.

3. THE LEO-D/T AND DS-T EXPERIENCES

The LEO terminal is the prototype for a new class of low cost ground station to reduce life cycle cost of tracking NASA missions in low earth orbit. The development was carried out in two phases by a small team of engineers and SeaSpace Inc., a US satellite ground terminal manufacturer. In the first phase, SeaSpace upgraded a COTS weather satellite-tracking terminal to receive downlink from NASA satellites. JPL provided the function of science data delivery via commercial circuits to the Principal Investigator (PI). This phase was completed in 1994, with successful one-week demonstrations of the automated, unattended operations tracking two NASA science satellites (SAMPEX in July and EUVE in December), both operated by NASA's Goddard Space Flight Center.

The second phase, command uplink capabilities were added to the prototype by JPL. A weeklong demonstration of the unattended uplink and downlink operation of the LEO-T with COBE satellite was completed in December 1995. Analysis of the terminal logs and received data indicated that the terminal operated flawlessly during the one-week demonstration. Following the validation demonstration, the terminal was left in unattended mode to track and receive telemetry data with the objective of collecting long-term reliability statistics. Operating during the next 26 months the terminal logged 3120 hours of tracking with one system malfunction (antenna controller circuit board failure).

After initial setup the terminal auto connects and retrieves orbital elements from the Naval Space Surveillance Center for each satellite. Based on the orbital elements the terminal automatically generates satellite view periods, antenna pointing and frequency predicts. The auto scheduler uses the view periods and user defined tracking priorities to track multiple satellites. For every scheduled pass the scheduler wakes up the terminal two minutes before Beginning Of Track. The terminal executes the automated, unattended pre/in/post-pass uplink/downlink routines and then waits for the next pass.

The LEO-T is built entirely from commercially available components. It uses a 3-meter aluminum mesh antenna enclosed in a fiberglass radome plus a 1.2-m rack for the electronics. With the 3-m dish the terminal is capable to support up to 55% of NASA's current and planned LEO missions; by using a 5-m dish the coverage increases to 70%. The radome protects the microwave electronics and the antenna from environmental conditions. Housing the 200 W S-Band solid-state transmitter in the radome base reduces RF loss in the cable running to the antenna feed. In the current configuration the terminal operates at NASA S-Band (up/down-link). Throughput for telemetry 1.2 Mb/s, command 2 Kb/s; the telemetry rate can easily modified by replacing the appropriate COTS modules. The terminal provides TCP/IP interfaces with remote users over commercial communication links.

The availability of commercial subsystems for the building of reliable automated ground stations provides the enabling environment for realizing the goals quickly and cost effectively. In particular, the automation, integration, and networking capabilities available with modern workstations provide the opportunity for a major paradigm shift: simply to view the ground station as a workstation that happens to have specialized hardware (antenna, receiver, etc.) as its peripherals.

The DS-T concept followed and leveraged on the autonomous, unattended operations model of the LEO-T. Modifications were necessary to address the inherent differences between requirements for ground station support of deep-space missions as compared with LEO spacecraft. These include: (1) Longer track times for deep-space missions means more complex sequences of events during a track. (2) While all of the LEO-T subsystems were COTS, because of the need for significantly improved performance, the DS-T uses subsystems specially designed for deep-space applications. (3) The custom-built deep-space subsystems work much closer to the theoretical limits than do the LEO subsystems, these are built-in limited numbers and generally do not enjoy trouble-free operations to the extent available with COTS equipment. This required a more capable error-detection and recovery algorithm in the ground station. (4) JPL subsystems have DSN-specific interfaces that would have encumbered the contractor if DS-T were to be built as a turnkey procurement. These constraints suggested a teaming arrangement between JPL and industry to leverage on the strengths of each party. We achieved our goal by leveraging the COTS ground-station-operations software complemented with a JPL scheduling component and dynamic script generation, resulting in cost-effective prototype development and the use of COTS components when appropriate.

DS-T's goal was a low-cost, quick demonstration of the proof of concept using a DSN antenna and NASA deep-space spacecraft, fully automated and autonomous (receive only) operations over several days. The demonstration covered: (1) Schedule-driven operation (only a high-level request was necessary), automated scheduling and conflict resolution within the DS-T. (2) Self-generated predicts for antenna pointing and receiver frequency information. (3) Automatic pre-track configuration and self-test. (4) Autonomous operation, with active monitoring of the track and with built-in error recovery. (5) Automatic post-track telemetry and monitor-data delivery to the PI. At Deep Space Station 26 (DSS 26, a 34-m Beam Wave Guide antenna located at Goldstone, California) the DS-T was successfully demonstrated with a single spacecraft, single-track capability in April 1998 and with multi-spacecraft and multi-track capability in lights-out mode over several days in September 1998 with Mars Global Surveyor (MGS). On 17 September 1998, the result of an error in the command sequence put the spacecraft in safe-hold mode, and normal operations were suspended. The 6-day lights-out mode demonstration had to be terminated after the first three successful days.

The DS-T used the DSS-26 BWG antenna. The antenna pointing and microwave component configurations were done by standard DSN components, used without any modification. The safety plan developed for automated, unattended operation required a systematic evacuation of the antenna site and the activation of a perimeter monitoring circuit. An interruption of the perimeter circuit stopped the antenna immediately. To have the necessary performance a DSN standard Block V Receiver was used as the primary receiver. A COTS receiver was tested, but not used because of insufficient link margin at higher data rates. Telemetry processing was done by a COTS package, without modifications, supplied by Avtec. The maximum throughput rate was 25 Mb/s. Each of the data transformation steps generated real-time-quality information and annotation of the telemetry data. A Sun Ultra-2 computer, running Solaris operating system, was used for the DS-T control at DSS 26. Lower performance workstations were used for the remote controller and PI position at JPL. For the task duration the connection between the equipment at DSS 26 and at JPL the remote-control and PI positions was via secure link over the NASA Science Internet connection using network encryption units. The largest software element in DS-T was the commercial EPOCH 2000 software (~ 120,000 lines of code) that provides the monitor and control (M&C) platform. It provides database-driven functionality for the automation and allows M&C from multiple locations. About 150,000 lines of existing debugged code developed at JPL through technology development programs were reused. The first major block of reused code is the predict generator; second, the DS-T schedule/automation engine is a version of the Demand Access Network Scheduler (DANS), and third, a modified version of the Automated Scheduling and Planning Environment (ASPEN) [9] for dynamic control-script generation. Customized software for the DS-T demonstration is estimated to be from 25,000 to 30,000 lines of code.

Three functional layers can be found within the DS-T software. From the top, these are the automation layer, the application layer, and the subsystem layer. The automation layer is responsible for the high-level control and execution monitoring of the station. It also provides the user interface to the autonomous station/terminal. Through the automation layer service requests are submitted to the system and then scheduled for execution. The application layer is for monitor and control; it is responsible for low-level control of the antenna track, the subsystems in the layer below as well as for the logging and archiving of relevant monitor data. It also provides the real-time operator interface, if requested, which displays monitor data and accepts low-level operator directives. Use of real-time operator directives is not necessary for the execution of a track, but it is available. The subsystem layer is made up of an interface and the subsystems themselves. In conjunction with the hardware mentioned above additional peripheral subsystems, such as a weather station, make up the lowest layer of the architecture. The layered approach provides several benefits. Most importantly, it provides clean boundaries between different functionalities in the system. This, combined with the proper abstraction levels, makes partitioning the system simpler, aiding in division of development and testing responsibilities.

The lessons learned from the DS-T task are that DSN antennas can be safely operated in an automated, schedule-driven mode. State of the art in ground station automation has reached the point where station operation can be left to autonomous unattended stations running on high-level directives [7]. Using automated procedure-generation techniques, the pre-track-generated dynamic scripts allow error recovery for a substantial class of real-time errors. However, based on the DS-T experience, a better approach is to generate a success-oriented script with the ability to generate and transfer control to a new error-recovery script as anomalies develop. This way the overall script is simpler, and unplanned interactions between various error-recovery routines do not interfere, while the execution is limited to one routine that resolves the cause of the problem.

4. DSSC ARCHITECTURE

The DSSC architecture was designed as an automation framework independent of the control problem to be resolved and we designed the architecture to scale. Collaboration took place between the DSSC team and the MDS (Mission Data Services) team at JPL during the design phase. MDS is an architectural effort at JPL intended to develop architecture for the future such that significant portions of a spacecraft design can be reused across different missions. This differs from the past where each spacecraft, for the most part, has been designed from scratch; making use of lessons learned from previous missions.

The architecture in Figure 2 shows the functionality required regardless of whether you are commanding something manually or autonomously. In order to perform a task reliably one must:

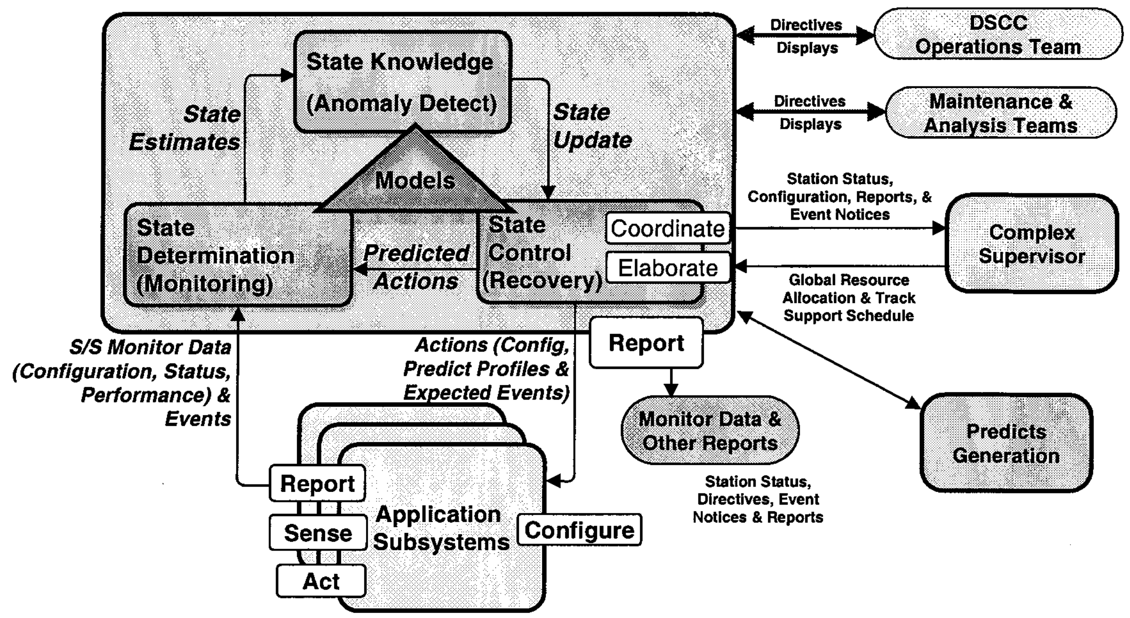
1. Decide on the actions to be taken to achieve the desired goal. This is a State Controller function
2. Convert into commands the desired actions, which are sent to the entity being controlled
3. Communicate desired goals to State Determination component for comparison in action 5
4. Monitor status information from the controlled entity to perform State Determination
5. Compare the predicted state, derived from the predicted action sequence, and the actual state as determined from the status information to develop State Knowledge or Anomaly Detection
6. Close the control loop by giving the status updates to the State Controller, which reassesses the next action to be taken that moves the system to the desired goal.

This cycle has been described as the Sense-Act-Plan or SPA cycle [10].

There is some functionality overlap in our instantiation of this architecture between the different tools utilized. At a certain level the planning and execution component performs all of the SPA tasks, but because of limitations to how planners perform in monitor and diagnostics functions, we utilize specific fault detection and isolation (FDI) techniques to provide better status information. For the planning and execution functionality we are utilizing a continuous planning paradigm provided by the Continuous Activity Scheduling Planning Execution and Re-planning (CASPER) system [5][6] combined with a task level control system execution functionality provided by Task Description Language (TDL). These two components have been combined into a single system for providing a framework for Closed Loop Execution and Recovery (CLEaR) [8].

For the FDI functionality we are utilizing:

- Beacon-Based Exception Analysis for Multi-missions (BEAM), which detects physical shifts in operational modes and identifies the contributing data sources, and
- Spacecraft Health Inference Engine (SHINE), which is a model driven rule-base system used to detect and isolate the source of an anomaly [11][12].



State Knowledge
(Anomaly Detect)
State Estimates
State Update
Models
State Determination (Monitoring)
Predicted Actions
State Control (Recovery)
Coordinate
Elaborate
Directives
Displays
DSCC Operations Team
Directives
Displays
Maintenance & Analysis Teams
Station Status, Configuration, Reports, & Event Notices
Complex Supervisor
Global Resource Allocation & Track Support Schedule
S/S Monitor Data (Configuration, Status, Performance) & Events
Actions (Config, Predict Profiles & Expected Events)
Report
Monitor Data & Other Reports
Predicts Generation
Station Status, Directives, Event Notices & Reports
Report
Sense
Act
Application Subsystems
Configure

Figure 2 DSSC Architecture Diagram

The reason for combining these components is that SHINE can determine whether the mode shift that BEAM detected is nominal or an anomaly.

Through the combining of CLEaR, with BEAM and SHINE into single architecture we enable the system to be self-aware of both its intentions and its well being. In short, the system is capable of both intelligent understanding and intelligent reasoning.

5. CLEAR

CLEaR is a hybrid controller utilizing Artificial Intelligence, continuous planning and executive techniques. CLEaR is built on top of the CASPER continuous planning system. CASPER provides the capability to perform command sequence generation, execution, monitoring and re-planning while enabling goal-based commanding of the entity, commanding at the level of what should be done. The 'how' is left to the system to decide depending on the given circumstances. CASPER's intent is to minimize the time required to produce/maintain a command sequence (plan) that is consistent with the goals, operating constraints and state. There are circumstances that require an event-driven reactionary mode of commanding; not valid for CASPER. To accommodate this need, we have increased the executive capabilities of CASPER by integrating TDL into CLEaR.

TDL plays several roles in increasing the execution capabilities: 1) Adds an event-driven sequencing capability that enables planning activities to be further expanded at execution time based on a procedural representation. 2) Adds a reactionary planning capability to accommodate stimulus/response type of situations where command sequences can be extended or modified in a rapid fashion. This is particularly useful for prompt responses to error conditions.

In our approach, CLEaR takes in a set of goals, abstract description of what needs to be done, and produces a plan which describes how the goals should be carried out. This is done utilizing an iterative repair-planning algorithm to reconstruct or repair the plan. As time advances and approaches the planned start time of an activity, it is marked for execution. The TDL component continues the execution process of the planner. This step may involve further expansion of the activity or simply results in the translation of the activity into the operational directives to be sent to the appropriate subsystem or controller. To ensure that the planner does not try to modify a portion of the plan that has already started executing, a commit window strategy has been adopted. Commit window applies also when the planner is still trying to modify the portion of the plan that needs to start executing. All activities whose start times are less than some time delta after the current time, are locked, or said to be inside the commit window, and will be executed. The commit window enables us to make soft-real-time claims that the planner will not be stuck in a planning cycle when it is time for an activity to execute. Any plan inconsistencies that occur within the commit window prior to execution (unresolved conflicts in the plan), or which occur as a result of execution (exceptions), must be handled by the reactive planning or error-recovery capabilities of TDL.

As TDL carries out the execution of the activities selected by the planner, the system's state is monitored and fed back into the TDL for analysis, which reacts to error conditions. This state information is used to update the predicted status of the system and the condition of the plan, then the planning component resolves if the plan needs modifying. Conditions that can be handled by TDL's reactive planning will not be seen as failures and as such the planner will not attempt to modify the plan. If TDL cannot resolve the issue locally it would produce a failed-goal status message that would result in the planner attempting to resolve the problem by modifying the plan. This provides an ability to be cognizant of failure at multiple levels. A failure can be recognized by the reactive planning component as it carries out execution of sub-goals, but if unresolved the global planner can have cognizance of failure and re-plan appropriately.

For a more detailed description of CLEaR and the ongoing research of how to balance the long-term goal-based global reasoning of a traditional planning approach with the short-term event-driven local reasoning of traditional executives see [8]. As our reasoning is only as good as our understanding of the state of the system, we deploy more sophisticated state determination techniques to provide a greater understanding of the state of the system. BEAM and SHINE are used to pre-process the status of the system and consolidate the information to be used by the planning system.

6. BEAM/SHINE

BEAM is a new method for automatic system analysis. It was originally funded by TMOD for use as an autonomous spacecraft monitor, but since then it has been adapted to a broad variety of applications. BEAM is a computational architecture that contains a series of reusable software modules to perform real-time system characterization, fault detection, and impact analysis. The process is driven by data. BEAM can efficiently employ physical models, if they are available. Because there are so few other requirements, it can be tailored rapidly to nearly any instrumented system. BEAM operates by studying regularly sampled performance data in real-time. This data is studied singularly, where individual signals are compared against their invariant properties, and in combination, where the sum total of sensor input is fused and system behavior as a whole is studied. To characterize the overall system behavior, a complex transform is applied to multiple signals simultaneously, yielding a single, evolving object called a Coherence Plot, which represents the system status. This object, which can be interpreted visually as well as mathematically, is tracked to reveal state transitions, emergence of faults and degradations, and actor signals or recognized types of failures. Once these global properties have been extracted, further analysis is done on the implicated signals in order to pinpoint sources of failure and progressions of incipient faults. A Coherence Plot contains a great deal of information and is in itself useful to a trained operator.

The transform also outputs a measure of System Stability. This is a single parameter extracted from the Coherence behavior over time representing the mode behavior of the total system. While these plots contain a wealth of information, they can be refined much further in order to produce an unambiguous and clear system assessment. Following the raw computation of the system coherence, differences between the computed and expected coherence, encapsulated in the "Difference Plot", are considered at mode transitions or upon significant deviation from nominal coherence (from training results). The coherence difference is localized to signals that participate in the transition; if the difference is unexpected or unrecognized this represents the fault.

The results from separate signal analysis methods are combined and interpreted using SHINE. Combination of these methods allows detection of complicated anomalies and the ability to distinguish between fault source signals and secondary effects, permitting a more sophisticated method of event recognition at the single signal level.

SHINE is a multi-mission, reusable knowledge base software tool for the monitoring, analysis and diagnosis of spacecraft and ground systems through forward and backward inference. SHINE advances the state-of-the art in artificial intelligence by solving a broad class of problems that previously were considered intractable because of real-time system requirements, high-speed and small size constraints. SHINE introduces a novel paradigm for knowledge visualization and ultra-fast inferencing that goes well beyond traditional forward and backward chaining methodology. A sophisticated mathematical transformation based on graph-theoretic data flow-analysis is introduced that reduces the complexity of conflict-resolution during the match cycle from $O(n^2)$ to $O(n)$ for many kinds of pattern matching operations. Computational overhead is further minimized by the built in source-to-source transformational system for the optimization of code generated from the rules through data flow reduction.

In order to detect modeled and un-modeled events and perform system prognostics SHINE uses real-time inputs and makes use of multiple types of knowledge such as detection, diagnostic, simulation and causal modeling. To combine these different approaches heuristic, experiential knowledge is used to quickly isolate candidate faults and then use deeper causal model-based reasoning to analyze the problem in detail and eliminate incorrect hypotheses. The Symbolic Data Model of SHINE is a knowledge-based system that provides a control structure that can easily switch between different types of knowledge and reasoning strategies. Also, it provides multiple knowledge representations suited to each type of knowledge employed and allows moving easily from one representation to the next during problem solving.

Part of our system uses an approach based upon DRAPhys (Diagnostic Reasoning About Physical Systems) developed at NASA Langley Research Center by Dr. Kathy Abbott [13][14][15]. One advancement over Abbott's system is that we include knowledge-based modules for specific strategies of diagnostic reasoning that does not need tight coupling. This makes construction of the expert system much easier because software and domain knowledge can be reused for models of different hardware. Like Abbott's system, we include a knowledge-based model that preprocesses the qualitative interpretation of sensor data prior to analysis.

The input is quantitative sensor data from either a real-time data source or archived data. This can take the form of a real-time data stream, or a "beacon" approach using only significant events. The fault monitor compares the sensor data with the output of the quantitative model that simulates the normally functioning physical system. The monitor signals a fault when the expected system state derived from the system model differs from the actual system state. The expected behavior is a combination of engineering predictions from the Gray Box physical model representation and real-time sensor values. When a fault is detected, the monitor provides the diagnostic process with a set of the abnormal sensor values in qualitative form (*e.g.* the symbol SNR is exceeding predictions with respect to current ground system configuration) along with time tags to show the temporal ordering of symptoms. The diagnostic process is divided into several discrete stages and each stage has a unique diagnosis strategy; see Figure 3.

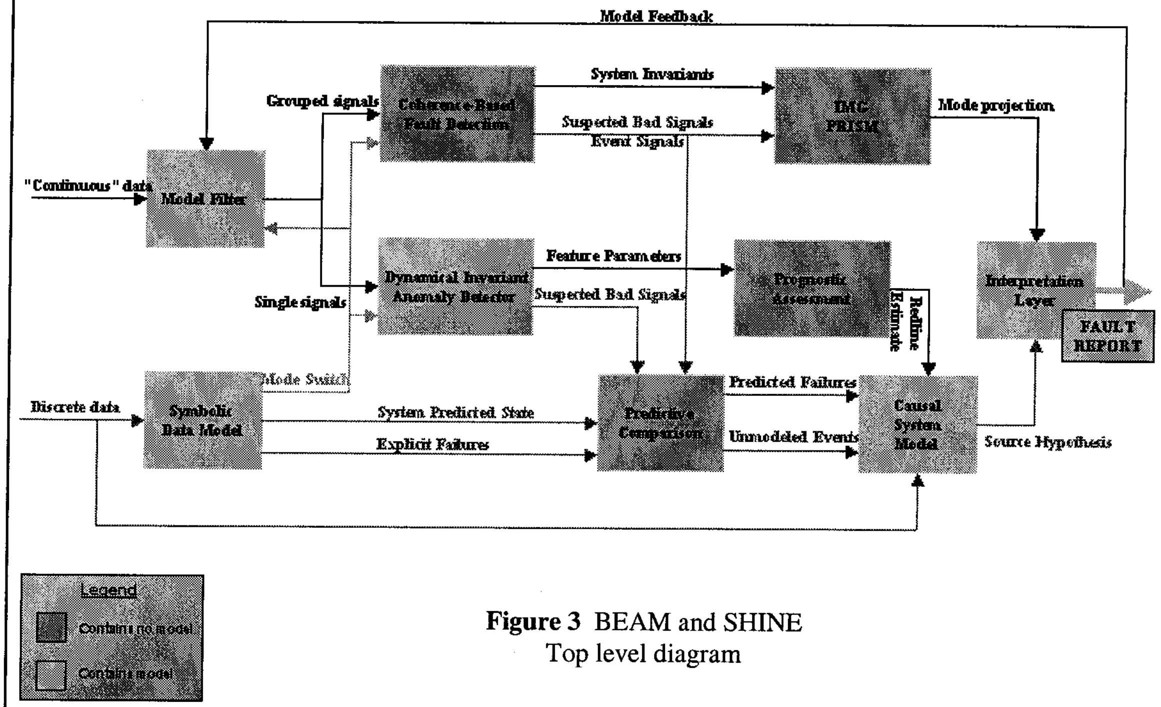


Figure 3 BEAM and SHINE
Top level diagram

In 1999, BEAM & SHINE were successfully demonstrated in the DSN as the FDI subsystem. We used BEAM to provide a framework of fault tolerance for key tracking observables such as SNRs and range frequencies. BEAM was also used as a prognostic tool by indicating when the system was deviating from the nominal performance requirements. It detected faults in parallel with SHINE. BEAM's output consisted of: 1) A normage value, which is an instantaneous estimate of the system as a single event based metric of system health 2) A normage limit, which is an instantaneous estimate of the system threshold 3) A ranked list that identified the channels associated with significant system behavior 4) An Operating map that is a summarization of the system channel behavior and provides a means of dynamic system state visualization. SHINE performed further fault identification and isolation using heuristic knowledge. Forward chaining rules were used to define semantics of FDI messages and to analyze telemetry frame data. Each data item is associated with one or more hypotheses that are generated as data arrives. Backward chaining rules were used to resolve all ambiguities in the hypotheses generated during telemetry frame collection. Model-based reasoning was used to combine real-time channel data with conclusions generated by the backward reasoning phase, and to map conclusions to actual hardware configuration.

The FDI subsystem participated in a DSN tracking of Near Earth Asteroid Rendezvous (NEAR) mission using the new Full Spectrum Array subsystem at the Goldstone Deep Space Communications Complex. The approach described above utilized BEAM and SHINE as a separate DSN sub-system providing FDI functionality to assist the operator in analysis of system performance. The current work is to utilize this capability to feed back into the CLEaR reasoning system and provide autonomous, lights-out ground station operations for a very large percentage of the tracks. In case of an exceptionally complex problem, this tool will help the expert human operators to solve it and to improve overall tracking performance with reduced staffing.

7. Conclusion

In this paper we have described the history of automated ground stations in the DSN, the current work on software control architecture being developed for the DSSC to provide ground station automation for NASA's Deep Space Network. We have described the specific technology components used to instantiate this architecture. Finally, we provided insight on how they fit together in the solution of a general-purpose software controller for autonomous operations in a number of domains.

8. Acknowledgements

The Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration performed this work. The Mission Services Element of JPL's Telecommunications and Mission Operations Technology Program supported portions of this work.

References

[1] Deep Space Network, Jet Propulsion Laboratory Publication 400-517, April 1994.

[2] F. Fisher, D. Mutz, T. Estlin, L. Paal, and S. Chien, "The Past, Present and Future of Ground Station Automation within the DSN," Proceedings of the 1999 IEEE Aerospace Conference, Aspen, CO, March 1999.

[3] N Golshan, W. Rafferty, C. Ruggier, M. Stockett and M. Wilheim "Low Earth Orbiter Demonstration Terminal" TDA Progress Report 42-125, May 1996.

[4] L. Paal, N. Golshan, F. Fisher, E. Law, W. Veruttipong, and M. Stockett "Deep Space Terminal Demonstration" TDA Progress Report 42-138, August 1999.

[5] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Integrated Planning and Execution for Autonomous Spacecraft," Proceedings of the 1999 IEEE Aerospace Conference, Aspen, CO, March, 1999.

[6] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling," Artificial Intelligence Planning Systems, Bolder, CO, April 2000.

[7] F. Fisher, S. Chien, L. Paal, E. Law, N. Golshan, and M. Stockett, "An Automated Deep Space Communications Station," Proceedings of the 1998 IEEE Aerospace Conference, Aspen, CO, March 1998.

[8] F. Fisher, R. Knight, B. Engelhardt, S. Chien, N. Alejandre, "A Planning Approach to Monitor and Control For Deep Space Communications," Proceedings of the 1998 IEEE Aerospace, Big Sky, MT, March 2000.

[9] A. Fukanaga, G. Rabideau, S. Chien, and D. Yan, "Toward an Application Framework for Automated Planning and Scheduling," Proceedings of the 1997 International Symposium of Artificial Intelligence, Robotics and Automation for Space, Tokyo, Japan, July 1997.

[10] E. Gat, "On Three-Layer Architectures," Artificial Intelligence and Mobile Robots – Case Studies of Successful Robot Systems, Kortenkamp, Bonasso and Murphy, AAAI Press, 1998

[11] James, Mark L. SHINE 5.7.4 Reference Manual (JPL Internal document) August 16, 1998.

[12] James, Mark L. and Atkinson, David J. "Software for Development of Expert Systems," NASA Tech Brief Vol. 14, No. 6, Item #8 from JPL Invention Report NPO-17536/7049, June 1990

[13] Abbott, Kathy H. Strategies and Representations for Onboard Aircraft Fault Diagnosis. SIGART Newsletter. Association for Computing Machinery. No. 92 April 1985a.

[14] Abbott, Kathy H. Exploration of Expert Systems Concepts for Onboard Diagnosis of Faults in a Turbofan Aircraft Engine. Proceedings of the American Control Conference. Boston, MA. 19-21 June 1985b.

[15] Abbott, Kathy H. Using Dynamic Behavior of Physical Systems for Real-time Fault Diagnosis: An AI Approach. IEEE Transactions on Systems, Man, and Cybernetics: Special Issue on Diagnostic Strategies. (Draft copy) 1986.